LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-222859

Implicit solvers for large-scale nonlinear problems

D. E. Keyes, D. Reynolds, C. S. Woodward

July 14, 2006

SciDAC PI Meeting
Denver, CO, United States
June 25, 2006 through June 29, 2006

Implicit solvers for large-scale nonlinear problems

David E. Keyes[1], **Daniel R. Reynolds**[2] **and Carol S. Woodward**[3, 4]

[1] Department of Applied Physics and Applied Mathematics, Columbia University, MC 4701, New York, NY 10027, USA

[2] Department of Mathematics, University of California at San Diego, 9500 Gilman Dr., La Jolla, CA 92093-0112, USA

[3] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, USA

E-mail: david.keyes@columbia.edu, drreynolds@ucsd.edu, cswoodward@llnl.gov

Abstract. Computational scientists are grappling with increasingly complex, multi-rate applications that couple such physical phenomena as fluid dynamics, electromagnetics, radiation transport, chemical and nuclear reactions, and wave and material propagation in inhomogeneous media. Parallel computers with large storage capacities are paving the way for high-resolution simulations of coupled problems; however, hardware improvements alone will not prove enough to enable simulations based on brute-force algorithmic approaches. To accurately capture nonlinear couplings between dynamically relevant phenomena, often while stepping over rapid adjustments to quasi-equilibria, simulation scientists are increasingly turning to implicit formulations that require a discrete nonlinear system to be solved for each time step or steady state solution. Recent advances in iterative methods have made fully implicit formulations a viable option for solution of these large-scale problems. In this paper, we overview one of the most effective iterative methods, Newton-Krylov, for nonlinear systems and point to software packages with its implementation. We illustrate the method with an example from magnetically confined plasma fusion and briefly survey other areas in which implicit methods have bestowed important advantages, such as allowing high-order temporal integration and providing a pathway to sensitivity analyses and optimization. Lastly, we overview algorithm extensions under development motivated by current SciDAC applications.

1. Introduction

The ability to solve large-scale, fully coupled multiphysics models is vital to progress in computational science; when it is lacking, various compromises are often made that cost time, scientific opportunity, or confidence in the results of a simulation. Efficient computation of solutions to such models requires robust and efficient algorithms for solving very large systems of coupled nonlinear algebraic equations. Particularly needed are algorithms that can exploit distributed hierarchical memory computers and have convergence rates that do not degrade as resolution improves or concurrency increases. A family of such algorithms has emerged in recent years; however, there is no universal best approach for all systems. The combinatorial and multiparametric richness of such algorithms, though at first confusing, provides a means of adapting to different physical regimes and different computer architectures, and also to exploiting legacy solvers – as a component of a more comprehensive approach.

[4] Work of this author was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Implicit nonlinear solvers potentially offer three primary benefits to the designer of a large-scale simulation code: affordable stability in multirate problems, affordable accuracy in complex problems, and (as a result of the way they are usually implemented) affordable sensitivities in all problems. Sensitivities (the rate at which perturbations in selected input parameters of a model amplify changes in selected outputs) are, in turn, fundamental to many of the ultimate scientific or engineering purposes of creating a simulation code: quantification of uncertainty and various forms of optimization, including design, control, and parameter identification.

Many multirate problems in science and engineering are characterized by the presence of slowly varying phenomena that hold most of the energy and interest in the simulation (e.g., that travel at the group velocity) and rapidly varying phenomena that are unimportant to the results of interest (e.g., that travel at the phase velocity). Examples abound: Alfven waves in a plasma discharge model, surface gravity waves in a climate model, acoustic waves in flight aerodynamics, fast chemical reactions among intermediates in complex biological or industrial systems. In these and many other systems, an explicit integration scheme is obligated to resolve the most rapid phenomenon in order to preserve numerical stability, as explicated by CFL theory, even though accurate resolution of the phenomena of interest could be achieved with time steps many orders of magnitude larger. CFL-type stability limits prevent even weak-scaling in the asymptotic limit of large numbers of available processors and therefore impose ultimate limits to resolution in practical computation. Therefore, the purpose of a simulation dictates the minimum time scale to be resolved; when it is substantially larger than the stability limit, implicit methods are called for. (We hasten to acknowledge that in many multirate systems, there are essential interactions between the disparate scales, and filtering the rapid transients may obscure critical phenomena, as in acoustically enhanced combustion, for instance. Implicit methods are not relevant for stability reasons alone to all problems, though robustness with respect to instability, especially instability whose cause is not undertood, is often the most dramatic motivation for implicitness.)

In the absence of effective nonlinear implicit methods, many complex problems are attacked by operator splitting, in which different terms of the governing system are advanced in different fractional steps, using optimized, process-specific solvers. Such splittings in first-order evolution equations generally result in first-order splitting errors (though higher-order splitting errors can be obtained in some important special cases). These errors thus render high-order temporal integration schemes for each component of the simulation irrelevant to overall simulation accuracy, and often impose a severe time step restriction due to accuracy or stability – up to orders of magnitude below an acceptable time step for a fully implicit integrator of the same or higher temporal order.

Efficient implicit nonlinear methods must generally be built around one of the numerous variants of Newton's method. Newton's method possesses quadratic local convergence hence it is ideally suited to any problem in which a good initial guess, or a good globalizer, is available. Transient problems come with good initial guesses for Newton's method, and many steady-state problems can be equipped with continuation schemes, using a physical or a discretization parameter, to quickly generate an iterate that lies within the radius of quadratic convergence. Newton is also well known to exhibit a convergence rate that is independent of spatial resolution in systems arising from PDEs, provided that this property can be achieved for the inner linear solver. The fundamental algorithmic challenge for Newton's method is to invert (merely approximately in many cases) the action of a Jacobian matrix on an arbitrary vector (see Section 2). This is precisely the action required to compute sensitivity information, where the vector is in this case not arbitrary, but consists of easily computed or estimated partial derivatives. Hence, many useful actions are built around the inverse action of a Jacobian matrix that otherwise require many expensive repeated forward solves of the governing system to reproduce.

We conclude that if an implicit method is available, it often becomes the solver of choice. It is therefore good news that an implicit method can often be constructed with relatively

modest machinery around an approximate or semi-implicit method, which is used as part of a preconditioner for the implicit, and sometimes from purely algebraic constructs that require little or no understanding of the governing system.

The balance of this chapter is organized as follows. Section 2 outlines the Newton-Krylov family of outer-inner iterative methods for nonlinear systems. Section 3 lists some of the freely available software, much of which originates within the U.S. DOE and is currently maintained and developed under the SciDAC program. Some new numerical results motivated directly by interactions between physicists and mathematicians under the SciDAC program in the area of plasma fusion are described in Section 4. Finally, Section 5 lists some of the challenges and opportunities facing algorithm developers in this area today.

2. Methods

As discussed above, implicit numerical approaches can provide a variety of benefits to the designer of a large-scale simulation code. Developing an implicit solution approach, however, can lead to a fairly complex structure of solvers. In this section, we will overview the most efficient nonlinear system solver family, its most common use, and some typical variations.

For time-dependent problems, or even for steady-state problems stabilized using continuation approaches, a typical implicit solution method will include an outermost time integration loop. At each step of this loop, an approximate solution will need to be calculated by solving a coupled, nonlinear discrete system. Let us denote this system as

$$F\left(u(t^n)\right) = 0, \tag{1}$$

where $u(t^n) \in I\!R^N$ is the vector of solution unknowns at time t^n, e.g. solution values, finite element weights, or even Fourier coefficients.

Due to its fast quadratic convergence and its ability to effectively use scalable preconditioners, the Newton-Krylov family of methods has become the standard for solvers of implicit, large-scale, nonlinear systems [1]. In this strategy, Newton's method is applied to solve the nonlinearities in the problem, and a Krylov iterative method is used to solve the linear systems arising within each Newton iteration. As Krylov methods can sometimes converge slowly, preconditioners are used to accelerate the linear solve. The main structure of this preconditioned Newton-Krylov solver applied to the system $F(u) = 0$ is as follows:

Assign an initial state u^0 at time t^0:
For each time step $n = 1, \ldots, N$ find $u^n \approx u(t^n)$ by Newton's method:

1. *Assign an initial Newton guess $u^{n(0)}$ (typically $u^{n(0)} = u^{n-1}$)*
2. *For each Newton iteration $k = 1, 2, \ldots$*
   A. *Using a Krylov method, approximately solve for s^{k+1},*
   $J_F\left(u^{n(k)}\right) s^{k+1} = -F\left(u^{n(k)}\right)$ *so that* $\left\| J_F\left(u^{n(k)}\right) s^{k+1} + F\left(u^{n(k)}\right) \right\| \leq ltol.$
   Each Krylov iteration requires:
      i. One matrix-vector multiply with $J_F\left(u^{n(k)}\right)$
      ii. One preconditioner solve
   B. *Update the Newton iterate, $u^{n(k+1)} = u^{n(k)} + \lambda s^{k+1}$*
   C. *Test for convergence, $\left\| F\left(u^{n(k+1)}\right) \right\| < ftol.$*

Here, $J_F\left(u^{n(k)}\right)$ is the Jacobian of the nonlinear function $F(u)$ evaluated at the previous Newton iterate $u^{n(k)}$, and $\lambda \in (0, 1]$ is a line search parameter chosen to help globalize the method.

Krylov methods develop an approximation to the solution of the linear system $Js = -F$ by iteratively building a Krylov subspace of dimension m defined by

$$\mathcal{K}(r_0, J) = span\left\{r_0, Jr_0, J^2 r_0, \ldots, J^{m-1} r_0\right\}, \tag{2}$$

where r_0 is the initial residual of the linear system. The approximation is then chosen depending on the particular Krylov method: a solution within the subspace that minimizes the linear system residual (a minimal residual method), or one that gives a residual orthogonal to the Krylov subspace (an orthogonal residual method). Further details of these methods can be found in [2, 3]. Within Newton-Krylov methods, the two most commonly used Krylov methods are GMRES [4] and BiCGStab [5], as these methods are designed to handle non-symmetric linear systems typical multi-physics simulation. GMRES tends to be the most used as it is very robust although it has a heavy memory requirement. While BiCGStab can have a much lower memory requirement, it is less robust as evidenced by a non-monotonically decreasing residual.

Newton's method is chosen for many applications because of its very fast convergence. Once the approximate solution, u^k is "close enough" to the true solution of the nonlinear system, u^*, then convergence is *q-quadratic*, i.e.

$$\|u^{k+1} - u^*\| \leq C\|u^k - u^*\|^2,$$

where C is a constant independent of u^k and u^*. This result assumes that the Jacobian systems are solved exactly. If these systems are solved inexactly, as in a Newton-Krylov method, then care must be taken to chose the linear system tolerance $ltol$ carefully in order to preserve convergence of the overall nonlinear method [6]. In particular, if we take $ltol = \eta^k \|F^k\|$, then q-quadratic convergence is retained if η^k is taken to be $C\|F^k\|$ so that $ltol = C\|F^k\|^2$, where C is a constant independent of the solution or current iterate. If instead, we have that $\lim_{k\to\infty} \eta^k = 0$, then convergence is provably q-superlinear, whereas if η^k is constant in k, convergence is only linear.

An issue often arising in use of Newton-Krylov methods is that of getting u^k "close enough" to the solution to see fast convergence. In order to improve the robustness and speed of the method, globalization techniques are often applied. These techniques take many forms, such as with line search methods, where the step in the Newton direction may be damped in order to ensure a sufficient decrease in F with a minimum step length. Other techniques include pseudo-transient continuation (for steady-state problems), where a false time stepping mechanism is added to aid solution while slowly increasing the time step to ∞; a hybrid nonlinear scheme, where one might start with a slower but more globally-convergent fixed point iteration and then switch to a Newton method as the solution is approached; or trust region methods, where directions other than the Newton direction are considered in order to more quickly find the solution. For more information on these and other techniques see [7].

The tolerance η^k of these linear system solves may be adjusted to aid efficiency of the overall method. Since the Newton system is a linear model of the original nonlinear system, the model is a better approximation as the solution of the nonlinear problem is approached. When far from the solution, then, one should not solve the linear model too precisely, and thus "oversolve" the step. As a result, Eisenstat and Walker introduced two choices for selecting these tolerances which take into account how well the nonlinear system is converging:

$$\eta^k = \frac{\left|\|F^k\| - \|J^{k-1}s^{k-1} + F^{k-1}\|\right|}{\|F^{k-1}\|}, \tag{3}$$

$$\eta^k = \gamma_1 \left(\frac{\|F^k\|}{\|F^{k-1}\|}\right)^{\gamma_2}, \tag{4}$$

where γ_1 and γ_2 are usually taken to be 0.9 and 2, respectively [8]. The first of these choices uses a measure of how well the linear model agreed with the nonlinear system at the prior step, while the second uses a measure of the rate of convergence of the nonlinear system. Eisenstat and Walker showed that under appropriate assumptions, these choices retain the local convergence of the underlying inexact Newton method [8].

One main advantage of Krylov methods for use within Newton's method is that they do not require formation of the Jacobian matrix. Instead, they only require matrix-vector products, that, for sufficiently differentiable $F(u)$, can be approximated by finite differences as given by

$$J_F\left(u^{n(k)}\right) v \approx \frac{F\left(u^{n(k)} + \sigma\right) - F\left(u^{n(k)}\right)}{\sigma} \tag{5}$$

while still preserving convergence of the overall method [9, 10]. Thus, as long as the nonlinear function can be evaluated at each linear iteration, the Newton-Krylov solve can proceed without forming derivatives or requiring memory for storage of the full Jacobian matrix.

One issue that arises within Newton-Krylov methods is the expense of performing these nonlinear function evaluations at each linear iteration. A recent variant on Newton-Krylov methods uses an approximation to the nonlinear function in the finite difference scheme given by

$$J_F\left(u^{n(k)}\right) v \approx \frac{\tilde{F}\left(u^{n(k)} + \sigma, u^{n(k)}\right) - F\left(u^{n(k)}\right)}{\sigma}, \tag{6}$$

where $\tilde{F}(v, w)$ is a related function such that $\tilde{F}(v, v) = F(v)$ and $\tilde{F}$ includes approximations to nonlinearities in F that are cheaper to evaluate than those found in the original problem. Under certain assumptions defined in [11] the resulting approach will again preserve the local convergence of Newton's method while reducing compute time over the original Newton-Krylov method.

In many instances, Krylov convergence can be slow, leading to poor performance of Newton-Krylov methods. For this reason, preconditioners are often applied to the linear iterations. The goal of preconditioning is to transform a linear system from one that is difficult for a Krylov method to solve to an easier one, solve the easier system, and transform the solution back to one for the original problem. Preconditioning can be formulated from the *right*, from the *left*, or from both. These can be viewed as the following systems, respectively.

$$\begin{aligned} (JP^{-1})(Ps) &= -F && \text{(right)}, \\ (P^{-1}J)s &= -P^{-1}F && \text{(left)}, \\ (P_1^{-1}JP_2^{-1})(P_2 s) &= -P_1^{-1}F && \text{(both)}. \end{aligned}$$

In practice, preconditioning amounts to finding a cheaper related system and inexactly solving this system within each linear iteration. We note that approximations employed within the preconditioner do not affect the overall accuracy of the solution to the nonlinear problem. For more details, see [2].

In the case of a full Newton method, the Jacobian matrix is formed, and incomplete or banded approximations to the Jacobian are factored and stored for use in preconditioning. Due to the directional derivative approximations (5), Newton-Krylov methods do not require explicit construction of the Jacobian matrix. As a result, more specific methods must be developed for preconditioning. In particular, preconditioning systems are formed through a number of approximation methods, such as operator splitting different physical phenomena and solving each operator in succession or using an approximate Jacobian such as would be formed

from a lagged or Picard iteration rather than a first order approximation such as in Newton's method. In situations when a simulation code is being converted from an explicit or semi-implicit formulation, these strategies often lead to systems that are computed and solved using functionality already in the simulation code. Thus, code developers can make use of current code capabilities which are already developed and verified.

3. Software

Due to the modular nature of the Newton-Krylov approach, there are a number of community-supported, high-quality software packages that implement one or all of the steps described in Section 2. In particular, the proposed TOPS center includes five packages with algorithms of use to implicit simulations. The SUNDIALS library implements the Newton-Krylov approach discussed above with line search globalization and Eisenstat and Walker tolerance selections. SUNDIALS also includes time integration packages for ordinary differential equations in CVODE and differential algebraic equations in IDA, along with extensions of these two integration packages for computation of sensitivity information [12]. The PETSC library enacts the above nonlinear framework, with additional support for handling parallel PDE-based systems, including performance monitoring capabilities. Additionally, if a code uses one of the PETSc-supplied data structures, a rich variety of preconditioning strategies including domain decomposition methods is also included [13]. The TRILINOS Project is comprised of a number of interoperable software packages that perform many scientific computation methods. Of applicability to nonlinear systems the TRILINOS NOX package of nonlinear solvers contains the Newton-Krylov approach as described above, along with a number of globalization methods and fixed point iterations. These implementations can be combined with the TRILINOS LOCA package, that enables continuation methods and bifurcation analysis [14].

A common theme throughout many of these software packages is the relative ease with which existing scientific application software may be expanded to use the Newton-Krylov approach. Through examination of the Newton-Krylov method, and specifically the use of the directional-derivative approximation (5), we see that the basic implicit solver framework relies on the ability to perform a relatively small number of operations: vector operations on u such as addition, norms and dot-products, along with application of the nonlinear residual function $F(u)$ that takes one vector as input (u) and returns a vector of the same size as output. Thus, the basic algorithmic requirements do not require knowledge of how data is laid out, only on the user's ability to supply these vector operations and residual evaluations. In many cases, vector operations may already exist for the data structures in use by a given application, in which case only the residual calculation need be supplied, and even that may often be constructed out of existing simulation routines. The three packages discussed above, SUNDIALS, PETSC, and TRILINOS, all implement the Newton-Krylov family of algorithms in terms of these basic operations. It is in this way that many applications may quickly begin to enjoy the benefits of increased stability and accuracy allowed through the Newton-Krylov approach.

Once the basic approach has proven successful, the major work in moving an application to larger scales is in the development of a scalable and effective preconditioner for the problem, and in ensuring that the function evaluation remains scalable. TOPS includes two other packages which, in addition to PETSC and TRILINOS, can provide preconditioning methods. The HYPRE project offers a number of high performance preconditioners with a focus on multigrid (structured and algebraic). This package supports a variety of discretization-based interfaces allowing a natural problem description for setting matrix data in the contexts of grids and stencils, finite elements, or algebraic for unstructured problems (with matrix and right-hand side) [15]. The SuperLU library supports the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines, and is based on LU decompositions and triangular solves [16]. Due to more invasive algorithmic requirements, these packages require

a user to place data into provided structures, however, in order to realize significant scalability.

We note that the computations in the following Section 4 were performed using the SUNDIALS library.

4. Numerical Examples

As described in Section 2, one significant benefit to using nonlinearly implicit approaches for integrating multi-physics simulations is that they are free from stability-based restrictions on the time step size. Instead, the time step may be chosen to provide temporally accurate simulations following the dynamical time scales of scientific interest. We illustrate this property with the following example from magnetic fusion simulations.

In magnetically-confined plasmas, magnetic reconnection (MR) processes convert magnetic field energy into plasma kinetic and thermal energy, resulting in large-scale topological changes in the magnetic flux surfaces that may interfere with or even halt the fusion process. Therefore, the ability to simulate such phenomena is critical to develop increased understanding and possibly control over reconnection events. One model that has been traditionally used to study these processes is that of *single-fluid resistive magnetohydrodynamics*, which couples the equations of hydrodynamics with resistive Maxwell's equations, modeling the plasma as a charged fluid that interacts with and on the background electromagnetic field. In this model, the speed of the reconnection process, or the *reconnection rate*, is of key interest, and depends on the magnetic resistivity η or Lundquist number S ($S \propto 1/\eta$). This dependence is known in the literature as *Sweet-Parker scaling* [17, 18], which states that the reconnection rate for unforced reconnection should scale proportionally to $S^{-1/2}$. Moreover, the reconnection process itself occurs in a thin current sheet, whose width also scales proportionally to $S^{-1/2}$. Therefore as the Lundquist number increases, the time-to-completion for MR simulations increases, while the spatial resolution required to resolve the current layer shrinks. Therefore, for explicit-time methods that must satisfy a diffusive CFL stability constraint in which $\Delta t \propto S\Delta x^2$, the computational time required to complete a simulation of the reconnection process should scale as $S^{3/2}$; similarly, semi-implicit methods that require time step constraints of $\Delta t \propto \Delta x$ should require simulation times that scale as $S^{3/2}$. The scale of the problem becomes apparent when we consider that the current state-of-the-art in simulation codes remains around $S = 10^4$. While this provides a realistic value for some magnetic fusion devices, such as CDX-U, it remains four to five orders of magnitude below the estimated values required to model next-generation fusion reactors, such as ITER (where $S \approx 5 * 10^8$).

As such, this problem proves an ideal example to demonstrate the benefits of using a fully-implicit approach, that, assuming perfectly-scalable solvers, should require computational times to complete MR simulations that scale as S. The results provided here use a high-order-accurate, fully implicit approach for solving the 2D single-fluid MHD equations, as described in [19], that is based on the CVODE solver from the SUNDIALS library [20, 21]. The results shown examine the computed time-to-simulation for this approach, compared with a similarly-accurate fully-explicit simulation code, as the Lundquist number is increased through the values $S = \{100, 500, 1000, 2500, 5000, 10000\}$. In these examples, the simulation times proceed just past the peak reconnection time, which we have used as $T = \{20, 40, 60, 90, 125, 180\}$. Additionally, as the Lundquist number is increased, we refine the mesh in order to resolve the current layer, using values of $\Delta x = \{0.4, 0.2, 0.1, 0.05, 0.025, 0.0125\}$. Moreover, as these meshes are refined, we increase the number of processors used so that $Np = \{1, 2, 4, 16, 64, 256\}$.

As seen in the plot for figure 1, both explicit and implicit simulations require an increasing number of time steps to complete the magnetic reconnection simulation as the Lundquist number is increased. However, while the explicit simulation begins requiring only 50% more time steps than the implicit at $S = 100$, this disparity grows to more than a factor of 5 at $S = 10000$, with the gap widening as S increases. We also see in Figure 2 that this improved scaling of

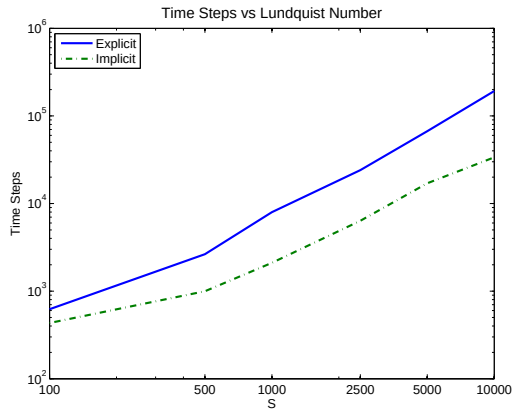

Figure 1. Explicit and Implicit time steps required for MR simulations with increasing S (and appropriate spatial mesh refinement).

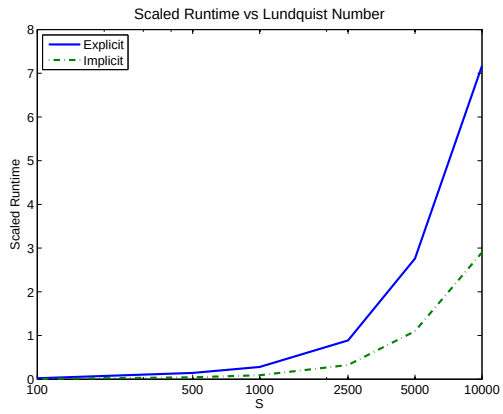

Figure 2. Explicit and Implicit scaled runtimes for magnetic reconnection simulations, as the Lundquist number is increased.

implicit time steps results in scaled run times that increasingly best those from the explicit simulation. By *scaled* run times, we show the simulation time required on average per spatial mesh point, which removes the increase in simulation time due to the refining spatial mesh from the comparison, in order to more accurately elucidate the effects of the time-stepping algorithm and solver scalability. We note, however, that as the Lundquist number is increased, and the spatial mesh is reduced to properly capture the thinning reconnection layer, the implicit solver must perform an increasing amount of work. This is due to the fact that as the mesh is refined, the condition number of the un-preconditioned Jacobian matrix grows as $1/\Delta x^2$. Thus, although the implicit solver requires fewer and fewer time steps compared with the explicit, those time steps are increasingly difficult to solve. As such, these non-preconditioned implicit results do not exhibit the optimal scaling of simulation time with S. However, even without the benefit of optimal preconditioners, these simple MR simulations already benefit tremendously from the use of the fully-implicit approach.

5. Looking Ahead

Despite the significant advances taken by applications such those described in Section 4, much research remains before Newton-Krylov methods can be used robustly in other situations. Specifically, open issues remain with respect to preconditioning, discontinuities, adaptive meshes, and variable constraints. These are all active areas of current research, and advances here will significantly increase the number of applications which can take advantage of larger, stable time steps tracking relevant physics.

As discussed above, preconditioners are required for many applications. The choice of such a preconditioner, however, is not immediately clear in some areas, and in others a fully scalable implementation may require significant effort. For example, in an application with nonlinear couplings between multiple species, the preconditioner may need to include couplings if they are tight, or may be able to split them apart. These types of choices are application-specific and must be handled with care.

Another difficulty with using the Newton family of methods is that they require continuity of the nonlinear function for provable convergence, and additional differentiability if directional derivative approximations (5) are used. Application of certain discretization schemes (such as limiters in fluid dynamics) or use of tabulated data (such as is found in diffusion coefficients in porous media flow) can result in discontinuous nonlinear functions. As a result, convergence

of these methods can stagnate in practice. Prior work has shown that careful selection of differentiable limiters in fluid dynamics can lead to effective Newton methods [22, 23], and use of spline functions can give robustness in situations with discontinuities in material data [24]. Much work remains, however, to determine the most effective discretizations for use in individual implicit formulations.

For adaptive meshes, there is no fundamental difficulty with applying implicit formulations, as the nonlinear problem can be viewed as existing on the final composite mesh that includes both coarse and fine regions. Issues arise, however, in assuring conservation properties hold, especially in the case of discretization schemes employing face- or node-centered unknowns. In these cases, explicit schemes more easily allow "refluxing" techniques which can distribute fluxes after the explicit update in order to assure conservation. These techniques are not straightforward within implicit approaches, as they may deleteriously affect solution accuracy as defined through the nonlinear residual function $F(u)$.

Lastly, many applications require variables to stay within a designated region (such as species concentrations remaining positive or energies not going negative). Typically, explicit codes include a post-timestep fix where variables are floored or moved within their allowed regions. Within an implicit solve as discussed above, the solver will update the solution and re-evaluate the nonlinear function repeatedly. As a result, difficulties may arise in computation of material parameters if solutions are moved along the Newton direction but out of their allowed regions. Methods for handling such constraints are included in most available software, though these methods tend to be *ad hoc*, and more robust techniques should be developed.

Acknowledgements
The authors wish to thank Ravi Samtaney for use of his magnetic reconnection code to generate the Sweet-Parker scaling example in Section 5.

References

[1] D. A. Knoll and D. E. Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *J. Comp. Phys.*, 193:357–397, 2004.
[2] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*, volume 1 of *Fundamentals of Algorithms*. SIAM, Philadelphia, PA, 2003.
[3] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. In A. Iserles, editor, *Acta Numerica*, pages 57–100. Cambridge Univ. Press, New York, 1992.
[4] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986.
[5] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.
[6] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.
[7] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, PA, 1996.
[8] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comp.*, 17(1):16–32, January 1996.
[9] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Statist. Comput.*, 11:450–481, 1990.
[10] P. N. Brown. A local convergence theory for combined inexact-Newton/finite-difference projection methods. *SIAM J. Numer. Anal.*, 24:407–434, 1987.
[11] P. N. Brown, H. F. Walker, R. Wasyk, and C. S. Woodward. On using approximate finite-differences in matrix-free Newton–Krylov methods. Submitted. LLNL Technical Report UCRL-JRNL-219284, 2006.
[12] SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic Solvers). http://www.llnl.gov/casc/sundials.
[13] PETSC (Portable, Extensible Toolkit for Scientific Computation). http://www-unix.mcs.anl.gov/petsc.
[14] Trilinos. http://software.sandia.gov/trilinos.
[15] HYPRE (High Performance Preconditioners). http://www.llnl.gov/casc/linear_solvers.
[16] SuperLU. http://www.nersc.gov/ xiaoye/SuperLU.

[17] D. Biskamp. *Magnetic reconnection in plasmas*. Cambridge monographs on plasma physics, third edition, 2000.

[18] J. Birn and et al. Geospace Environmental Modeling (GEM) magnetic reconnection challenge. *J. Geophys. Res.*, 106:3715–3719, 2001.

[19] D. R. Reynolds, R. Samtaney, and C. S. Woodward. A fully implicit numerical method for single-fluid resistive magnetohydrodynamics. To appear in *J. Comp. Phys.*, 2006.

[20] A. C. Hindmarsh and R. Serban. User documentation for CVODE v. 2.2.1. Technical Report UCRL-SM-208108, LLNL, Dec 2004.

[21] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31(3):363–396, 2005.

[22] V. Venkatakrishnan. Convergence to steady state solutions of the euler equations on unstructured grids with limiters. *J. Comp. Phys.*, 118:120–130, 1995.

[23] W. D. Gropp, D. E. Keyes, L. C. McInnes, and M. D. Tidriri. Globalized Newton–Krylov–Schwarz algorithms and software for parallel implicit CFD. *Int. J. High Performance Computing Applications*, 14:102–136, 2000.

[24] C. T. Miller, G. A. Williams, C. T. Kelley, and M. D. Tocci. Robust solution of Richards' equation for nonuniform porous media. *Water Resour. Res.*, 34:2599–2610, 1998.